Рубежный контроль №2 Студент: Попов М.Ю. Группа: ИУ5-25М

Решение задачи классификации текстов.

Необходимо решить задачу классификации текстов на основе любого выбранного Вами датасета (кроме примера, который рассматривался в лекции). Классификация может быть бинарной или многоклассовой. Целевой признак из выбранного Вами датасета может иметь любой физический смысл, примером является задача анализа тональности текста.

Необходимо сформировать два варианта векторизации признаков - на основе CountVectorizer и на основе TfidfVectorizer.

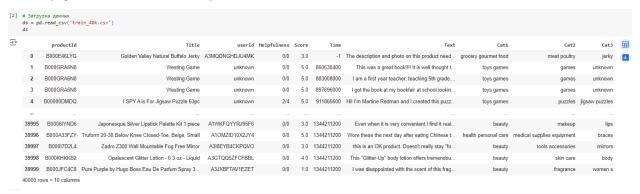
В качестве классификаторов необходимо использовать два классификатора по варианту для Вашей группы:

ИУ5-25M, ИУ5И-25M, ИУ5И-26M SVC LogisticRegression

Импортируем библиотеки

```
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import pandas as pd
import time
```

Загружаем набор данных



В качестве исходных данных был выбран набор данных 'train_40k.csv'. В нём некоторые признаки содержат пропуски:

```
🧹 [3] # проверим пропуски в данных и устраним их
    na_mask = ds.isna()
     na_counts = na_mask.sum()
      na counts

→ productId 0

     Title 16
userId 0
Helpfulness 0
      Score 0
      Time
                  0
      Text
      Cat1
      Cat2
Cat3
      dtype: int64
  ds.dropna(inplace=True)
     na_mask = ds.isna()
     na_counts = na_mask.sum()
     na_counts

→ productId 0

      Title 0 userId 0
      Helpfulness 0
      Score 0
      Time
      Text
      Cat1
      Cat2
Cat3
      dtype: int64
```

Разделим набор данных на обучающую и тестувую выборки

```
X, Y = ds['Text'], ds['Cat1']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

time_arr = []
# векторизация признаков с помощью CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_test_counts = count_vect.transform(X_test)
# векторизация признаков с помощью TfidfVectorizer
tfidf_vect = TfidfVectorizer()
X_train_tfidf = tfidf_vect.fit_transform(X_train)
X_test_tfidf = tfidf_vect.transform(X_test)
```

Часть 1

Произведем обучения вдух классификаторов (по варианту) для CountVectorizer

```
# SVC
gbc = SVC()
start time = time.time()
gbc.fit(X train counts, y train)
train_time = time.time() - start_time
time arr.append(train time)
pred gbc counts = gbc.predict(X test counts)
print("Точность (CountVectorizer + SVC):", accuracy_score(y_test,
pred gbc counts))
# Logistic Regression
lr = LogisticRegression(max iter=1000)
start time = time.time()
lr.fit(X train counts, y train)
train time = time.time() - start time
time_arr.append(train_time)
pred lr counts = lr.predict(X test counts)
print("Точность (CountVectorizer + LogisticRegression):",
accuracy score(y test, pred lr counts))
```

Часть 2

Произведем обучения вдух классификаторов (по варианту) для TfidfVectorizer

```
# SVC
qbc = SVC()
start time = time.time()
gbc.fit(X_train_tfidf, y_train)
train time = time.time() - start time
time arr.append(train time)
pred gbc tfidf = gbc.predict(X test tfidf)
print("Точность (TfidfVectorizer + LinearSVC):", accuracy score(y test,
pred gbc tfidf))
# Logistic Regression
lr = LogisticRegression(max iter=1000)
start time = time.time()
lr.fit(X_train_tfidf, y_train)
train time = time.time() - start time
time arr.append(train time)
pred lr tfidf = lr.predict(X test tfidf)
print("Точность (TfidfVectorizer + LogisticRegression):",
accuracy score(y test, pred lr tfidf))
```

Сравним полученные результаты:

A		L
Модели +	Точность валидации	Время обучения
(TfidfVectorizer + LinearSVC)	0.834063	30.3454
(TfidfVectorizer + LogisticRegression)	0.833938	917.7
(CountVectorizer + LogisticRegression)	0.817932	581.446
(CountVectorizer + LinearSVC)	0.755658	108.9
T		

Вывод: Лучше всего показала себя пара TfidfVectorizer + LinearSVC